Vol.2020-AL-179 No.4
2020/9/1

A Unified Approach for Designing Succinct Navigational Oracles for Families of Intersection Graphs on Circle

Hüseyin Acan[1,a)] Sankardeep Chakraborty[2,b)] Seungbum Jo[3,c)] Kei Nakashima[4,d)]
Kunihiko Sadakane[4,e)] Srinivasa Rao Satti[5,f)]

Abstract: We consider the problem of designing succinct navigational oracles i.e., succinct data structures supporting basic navigational queries such as degree, adjacency and neighborhood efficiently for intersection graphs on a circle, which include graph classes such as *circle*, *k-polygon-circle*, *circle-trapezoid*, *trapezoid* graphs. We first prove a general lower bound for these intersection graph classes, and then present a uniform approach that lets us obtain matching lower and upper bounds for representing each of these graph classes. More specifically, our lower bound proofs use a unified technique to produce tight bounds for all these classes, and this is followed by our data structures which are also obtained from a unified representation method to achieve succinctness for each class.

Keywords: Succinct Data Structures, Intersection graphs, Counting lower bound

1. Introduction

Intersection graphs of geometric objects are fascinating combinatorial objects from the point of view of algorithmic graph theory as many hard (NP-complete in general) optimization problems become easy, i.e., polynomially solvable when restricted to various classes of intersection graphs. Thus, they provide us with clues with respect to the line of demarcation between P and NP, if there exists such a line. Furthermore, they also have a broad range of practical applications [26, Chapter 16]. Perhaps the simplest and most widely studied such objects are the interval graphs, intersection graphs of intervals on a line [14, 15, 17]. Several characterizations of interval graphs [15] including their linear time recognition algorithms are already known in the literature [16]. There exist many generalizations of interval graphs, and we focus particularly in this work on some of these generalizations involving intersection of geometric objects bound to a circle.

More specifically, we study *circle graphs*, *k-polygon-circle graphs*, *circle-trapezoid graphs*, and *trapezoid graphs* in this article. A circle graph is defined as the intersection graph of chords in a circle [5, 8]. *Polygon-circle* graphs [18] are the intersection graphs of convex polygons inscribed into a circle, and the special case, when all the convex polygons have exactly k corners, we call the intersection graph k-polygon-circle [7]. Circle-trapezoid graphs are the intersection graph of circle trapezoids on a common circle, where a circle trapezoid is defined as the convex hull of two disjoint arcs on the circle [11]. Finally, trapezoid graphs are the intersection graphs of trapezoids between two parallel lines which can be regarded as a circle with a sufficiently large radius. These graphs are not only theoretically interesting to study but they also show up in important practical application domains, e.g., in VLSI physical layout [15, 26]. In spite of having such importance and being such basic geometric graphs, we are not aware of any study of these aforementioned objects using the lens of *succinct data structures* [24] where we need to achieve the following twofold tasks. The first goal is to bound from below the cardinality of a set T consisting of combinatorial objects with certain property, and this is followed by storing any arbitrary member $x \in T$ using the information theoretic minimum of $\log(|T|) + o(\log(|T|))$ bits (throughout this paper, log denotes the logarithm to the base 2) while still being able to support the

[1] Drexel University, United States
[2] National Institute of Informatics, Japan
[3] Chungbuk National University, South Korea
[4] The University of Tokyo, Japan
[5] Seoul National University, South Korea
[a)] huseyin.acan@drexel.edu
[b)] sankardeep.chakraborty@gmail.com
[c)] sbjo@chungbuk.ac.kr
[d)] kei_nakashima@mist.i.u-tokyo.ac.jp
[e)] sada@mist.i.u-tokyo.ac.jp
[f)] ssrao@cse.snu.ac.kr

relevant set of queries efficiently on x, both the tasks we focus on here. We assume the usual model of computation, namely a $\Theta(\log n)$-bit word RAM model where n is the size of the input. This is a standard assumption that implies a vertex can be distinguished, in constant time, with a label that fits within a word of the RAM. Finally all the graphs we deal with in this paper are simple, undirected, unlabelled and unweighted.

1.1 Related Work

Succinct navigational oracles. There already exists a huge body of work on representing several classes of graphs succinctly along with supporting basic navigational queries efficiently. A partial list of such special graph classes would be arbitrary graphs [10], trees [22], planar graphs [2], chordal graphs [23], graph with bounded tree-width k (partial k-trees) [9], etc. Specially, one can consider (i) circular-arc graphs (intersection graphs on the arcs on a circle), (ii) interval graphs (sub-class of circular-arc graphs), and (iii) permutation graphs (intersection graphs of line segments between two parallel lines) as the special case of the intersection graphs on a circle. For interval graphs and circular-arc graphs, Gavoille and Paul [13] (and independently, Acan et al. [1]) showed that $n \log n - O(n \log \log n)$ bits are necessary for representing an interval or a circular-arc graph with n vertices. In [1], the authors also presented succinct navigation oracles for both graph classes. Also for permutation graphs, a lower bound of $(n \log n - O(n \log \log))$ bits is known [4, 19].

Algorithmic graph-theoretic results. All the intersection graphs that we focus in this paper are very well studied in the algorithmic graph theory literature. Circle graphs (which are essentially same as *overlap graphs*[*1]) can be recognized in polynomial time along with admitting polynomial time algorithms for various optimization problems like feedback vertex set and independent set (see [26] and references therein for more details). These graphs were first introduced in the early 1970s, under the name *alternance graphs*, as a tool used for sorting permutations using stacks [8]. The introduction of polygon-circle graphs (which are same as *spider graphs* [18]) was motivated by the fact that this class of graphs is closed under taking induced minors. Even though the problem of recognising polygon-circle and k-polygon-circle graphs is NP-complete [20, 25], many optimization problems that are otherwise NP-Complete on general graphs can be solved in polynomial time given a polygon-circle representation of a graph (see [26] for more details). Felsner et al. [11] introduced circle-trapezoid graphs as an extension of trapezoid graphs and devised polynomial time algorithms for maximum weighted clique and maximum weighted independent set problems. We refer the reader to [14, 15, 21, 26] for more details on these graph classes and other related problems.

[*1] https://www.graphclasses.org/classes/gc˙913.html

Table 1: Lower bounds of families of intersection graphs.

Graph class	Space lower bound (in bits)	Reference (this paper)
circle	$n \log n - O(n)$	Theorem 1.4
k-polygon-circle	$(k-1)n \log n - O(kn \log \log n)$	Theorem 1.1
circle-trapezoid	$3n \log n - 4 \log \log n - O(n)$	Theorem 1.1
trapezoid	$3n \log n - 4 \log \log n - O(n)$	Lemma 1

1.2 Our Results

In this paper, we consider a graph class defined as the intersection graphs of objects on a circle, where objects are *generalized polygons*, polygons whose corners are on the circle and edges are either chords or arcs of the circle. This contains many graph classes including (1) interval graphs, (2) permutation graphs, (3) circular-arc graphs, (4) *k-polygon-circle graphs*, which are intersection graphs of polygons on a circle, where every polygon has k chords, and (5) circle-trapezoid graphs. Note that these example classes correspond to k-polygon circle graphs with a fixed k, while our upper and lower bounds in fact apply to a more general case when the graph contains polygons with different number of corners.

We first show a space lower bound for representing the above general graph class (Theorem 1.1). These new lower bound results for representing such graph classes are summarized in Table 1. Note that from Theorem 1.1, we can also obtain the space lower bounds for representing permutation graphs and interval graphs, which match the current best lower bounds for these graph classes [1, 4, 13, 19]. Furthermore using a similar idea to prove the main theorem, we also obtain a space lower bound for representing trapezoid graphs.

Next, we consider data structures for representing families of intersection graphs on a circle which support three basic navigation queries efficiently, which are defined as follows. Given a graph $G = (V, E)$ such that $|V| = n$ and two vertices $u, v \in V$, (i) $\mathsf{degree}(v)$ query returns the number of vertices that are adjacent to v in G, (ii) $\mathsf{adjacent}(u, v)$ query returns true if u and v are adjacent in G, and false otherwise, and finally (iii) $\mathsf{neighborhood}(v)$ query returns all the vertices that are adjacent to v in G.

We give a unified representation of families of intersection graphs of generalized polygons on a circle where generalized polygon is define as a shape where every pair of consecutive corners are connected by either an arc or a chord on a circle. We summarize our result in the following theorems.

Theorem 1.1. *Consider a class of intersection graphs on a circle*

consisting of n polygons, each of which has at most k chords or arcs. Let n_i be the number of all polygons on the circle with i corners, $\bar{n} = (n_2, n_3, \ldots, n_k)$, and $N = \sum_{i=2}^{k} i \cdot n_i$. Let $P_{n,k,\bar{n}}$ denote the total number of such graphs. Then, the following holds:

$$\log P_{n,k,\bar{n}} \geq \sum_{i=2}^{k} n_i \cdot i \log \frac{n}{i} - n \log n - \mathrm{O}(N \log \log n).$$

Theorem 1.2. *Consider an intersection graph of n (generalized) polygons on a circle. Let n_i be the number of all polygons on the circle with i corners ($2 \leq i \leq k$), where k is the maximum number of corners among the polygons on a circle, and N be the total number of corners of the polygons. There exist a $\left(\sum_{i=2}^{k} n_i \cdot i \log \frac{n}{i} - n \log n + \mathrm{O}(N \log k)\right)$-bit representation of the graph that can support* adjacent(u, v) *query in* $\mathrm{O}(k \log \log n)$ *time, and* neighborhood(v) *and* degree(v) *queries in* $\mathrm{O}(k|\textsf{degree}(v)| \cdot \log \log n)$ *time. Also, the representation is succinct (i.e., the space usage is* $(1 + \mathrm{o}(1)) \log P_{n,k,\bar{n}}$*) when* $k = \mathrm{o}(\log n / \log \log n)$.

Corollary 1.3. *For an intersection graph of n (generalized) polygons with at most k corners on a circle, let N be the total number of corners of the polygons. There exist an* $((N - n) \log n + \mathrm{O}(N \log k))$*-bit representation of the graph. For a k-polygon-circle graphs, there exists a* $((k-1)n \log n + \mathrm{O}(nk \log k))$*-bit representation.*

From these results, we can obtain succinct data structures for all the graphs classes in Table 1 which can support adjacent, degree, and neighborhood queries efficiently. Note that for the above mentioned classes of graphs, these are the first succinct data structures.

Finally, for circle graphs, we show that this lower bound can be improved to $n \log n - \mathrm{O}(n)$ bits. Furthermore, for circle graphs and circle-trapezoid graphs, we present alternative succinct data structures which support faster degree queries (for vertices whose degree is $\Omega(\log n / \log \log n)$). We summarize the results in the following theorem and lemma.

Theorem 1.4. *Let G be an unlabeled circle (trapezoid resp.) graph with n vertices. Then*

(i) at least $n \log n - \mathrm{O}(n)$ *bits are necessary to represent a circle graph G; and*

(ii) there exists an $n \log n + \mathrm{o}(n \log n)$*-bit (*$3n \log n + \mathrm{o}(n \log n)$*-bit resp.) data structure representing G such that* degree(v) *and* adjacent(u, v) *query can be answered in* $\mathrm{O}(\log n / \log \log n)$ *time, and* neighborhood(v) *query can be reported in* $\mathrm{O}(|\textsf{degree}(v)| \cdot \log n / \log \log n)$ *time.*

Lemma 1. *Consider a family of intersection graphs made from n trapezoids on two parallel lines. Let P_n denotes the total number of such graphs. Then the following holds:*

$$\log P_n \geq 3n \log n - 4n \log \log n - \mathrm{O}(n).$$

Due to lack of space, we omit the proofs of Theorem 1.4, Lemma 1.

1.3 Paper Organization

After listing preliminary data structures that will be used throughout our paper in Section 2, we move on to present the central contributions of our work. In Section 3, we prove all the lower bound results mentioned in Table 1 and present our general upper bound result (see Theorem 1.2) that provides succinct data structures for all these graphs in a unified manner. Finally, we conclude in Section 4 with some open problems.

2. Preliminaries

In this section, we introduce some data structures that will be used in the rest of the paper.

Rank, Select and Access queries. Let $A[1 \ldots, n]$ be an array of size n over an alphabet $\Sigma = \{0, 1, \ldots, \sigma - 1\}$ of size σ. Then for $1 \leq i \leq n$ and $\alpha \in \Sigma$, we define the rank, select and access queries on A as follows.

- $\mathsf{rank}_\alpha(i, A)$ returns the number of occurrences of α in $A[1 \ldots i]$.
- $\mathsf{select}_\alpha(i, A)$ returns the position j where $A[j]$ is the i-th α in A.
- $\mathsf{access}(i, A)$ returns $A[i]$.

Then, the following data structures are known for supporting the above queries.

Lemma 2 ([6])**.** *Given a bit array $B[1 \ldots n]$ of size n, there exists an $n + o(n)$-bit data structure which answers* rank_α, select_α *for* $\alpha = \{0, 1\}$*, and* access *queries on B in $O(1)$ time.*

Lemma 3 ([3])**.** *Given an array $A[1 \ldots n]$ over $\Sigma = \{0, 1, \ldots, \sigma - 1\}$ for any $\sigma > 1$, there exists an $nH_0 + o(n) \cdot O(H_0 + 1)$-bit data structure that answers* rank_α *and* access *queries in $O(1 + \log \log \sigma)$ time and* select_α *queries in $O(1)$ time on S, for any $\alpha \in \Sigma$, where $H_0 \leq \log \sigma$ is the order-0 entropy of A.*

Range minimum and maximum queries. Let $A[1 \ldots, n]$ be an array of size n over a totally ordered set. Then for $1 \leq i \leq j \leq n$, we define the rmq, rMq queries on A as follows.

- $\mathsf{rmq}(A, i, j)$: returns the index m of A that attains the minimum value $A[m]$ in $A[i \ldots j]$. If there is a tie, returns the leftmost one.
- $\mathsf{rMq}(A, i, j)$: returns the index m of A that attains the maximum value $A[m]$ in $A[i \ldots j]$. If there is a tie, returns the

leftmost one.

Lemma 4 ([12]). *Given an array $A[1 \dots n]$ of size n over a totally ordered set, there exists a $2n + o(n)$-bit data structure which answers $\mathsf{rmq}(A, i, j)$ queries in $O(1)$ time.*

Note that the above structure does not access A at query time. Similarly, one can also obtain a $2n + o(n)$-bit data structure supporting range maximum queries in $O(1)$ time.

3. Unified Lower and Upper Bounds

In this section, we give a unified representation of families of intersection graphs of generalized polygons on a circle. We assume that an arc is adjacent to only chords, otherwise we can merge two consecutive arcs into one. Note that we define a single chord (or an arc) as a polygon with two corners. Since there is no restriction on the number of corners for each polygon, this graph is a generalization of circle, k-polygon-circle and circle-trapezoid graphs. We note that a circular-arc graph can be represented by an intersection graph of generalized polygons with one arc and one chord on a circle, because if a shape on a circle intersects the chord, it always intersects the arc.

3.1 General Lower Bounds

We start with proving the Theorem 1.1. Suppose that the circle polygon graph is given as a polygon circle representation with n polygons on the circle. We will consider partially-colored circle polygon graphs obtained from the following construction. Take $m \le n$ (to be determined) non-intersecting polygons $A_1, \dots, A_m$ and paint A_i with color i. Let the set of these m polygons be S. For each of the remaining $n - m$ polygons, we will choose a subset of S, and for each such subset X we will construct a polygon with $|X|$ corners such that distinct corners lie on distinct polygons from X. Note that each edge of such a polygon intersects with exactly two colored polygons. This construction gives us a polygon-circle graph with n vertices, where m of these vertices are colored and they form an independent set. For $2 \le i \le k$, let n_i and m_i ($\le n_i$) be the number of all polygons and colored polygons on the circle with i corners respectively, and let M be the number of total corners on the colored polygons. From the definition, it is clear that $n = \sum_{i=2}^{k} n_i$, $m = \sum_{i=2}^{k} m_i$, $N = \sum_{i=2}^{k} i \cdot n_i$, and $M = \sum_{i=2}^{k} i \cdot m_i$. Let $\bar{m} = (m_2, m_3, \dots, m_k)$. Let us denote by $C_{n,k,\bar{n},\bar{m}}$ the number of such colored polygon-circle graphs, and by $P_{n,k,\bar{n}}$ the number of uncolored polygon-circle graphs.

We can first obtain an inequality $\binom{n_2}{m_2}\binom{n_3}{m_3}\dots\binom{n_k}{m_k} \cdot m! \cdot P_{n,k,\bar{n}} \ge C_{n,k,\bar{n},\bar{m}}$ since every graph counted in $C_{n,k,\bar{n},\bar{m}}$ can be obtained by choosing and coloring m_i polygons from n_i polygons on its polygon circle representation of uncolored one for each $2 \le i \le k$. Now we will find a lower bound for $C_{n,k,\bar{n},\bar{m}}$, which in turn will give a lower bound for $P_{n,k,\bar{n}}$. Let us denote the collection of i-subsets of S by S_i. Hence $|S_i| = \binom{m}{i}$. Also let $\mathcal{S}$ be a set of all possible k-tuples $(Y_2, Y_3, \dots, Y_k)$ where Y_i is a $(n_i - m_i)$-subset of S_i. Then the total number of graphs obtained by the above construction is at least $|\mathcal{S}| = \binom{m}{2}_{n_2-m_2}\binom{m}{3}_{n_3-m_3}\dots\binom{m}{k}_{n_k-m_k}$ by the following observations:

(i) For any subset of $\mathcal{S}$ with cardinality $\sum_{i=2}^{k}(n_i - m_i) = n - m$, we get at least one graph. (We might get more as the relative order of the corners of polygons within one colored polygon matters.)

(ii) If $\mathcal{T}_1$ and $\mathcal{T}_2$ are subsets of $\mathcal{S}$ and $\mathcal{T}_1 \ne \mathcal{T}_2$, then no matter how the corners of the polygons in $\mathcal{T}_1$ and $\mathcal{T}_2$ are chosen, the graphs corresponding to these two subsets will be different. Basically, in the graphs obtained from this construction, uncolored $n - m$ vertices are distinguishable by only looking at their colored neighbors.

Now we obtain the lower bound of $\log P_{n,k,\bar{n}}$ as follows. From the above arguments, we obtain $C_{n,k,\bar{n},\bar{m}} \ge \binom{n}{n_2-m_2}\binom{n}{n_3-m_3}\dots\binom{n}{n_k-m_k}$. Combining with the upper bound of $C_{n,k,\bar{n},\bar{m}}$, we obtain

$$\log P_{n,k,\bar{n}} \ge \sum_{i=2}^{k} \log \binom{\binom{m}{i}}{n_i - m_i} - \sum_{i=2}^{k} \log \binom{n_i}{m_i} - \log m!.$$

We set $m = \frac{n}{\log n}$. For each term of the right-hand side, the following inequalities hold by $\binom{a}{b} \ge (a/b)^b$.

$$\sum_{i=2}^{k} \log \binom{\binom{m}{i}}{n_i - m_i} \ge \sum_{i=2}^{k} \log \left(\frac{\binom{m}{i}}{n_i - m_i} \right)^{n_i - m_i}$$
$$\ge \sum_{i=2}^{k} n_i \cdot i \log \frac{n}{i} - M \log n - N \log\log n - n \log n$$

$$\sum_{i=2}^{k} \log \binom{n_i}{m_i} \le \sum_{i=2}^{k} m_i \log n = m \log n \le n$$

Therefore,

$$\log P_{n,k,\bar{n}} \ge \sum_{i=2}^{k} n_i \cdot i \log \frac{n}{i} - M \log n - N \log\log n - n \log n - \mathrm{O}(n)$$

To satisfy $M \le N/\log n$, we choose (and color) m polygons as follows. For $1 \le j \le n$, let d_j be the number of corners of j-th polygon in the representation. Without loss of generality, we order the polygons to satisfy $d_1 \le d_2 \le \dots \le d_n$. Now we claim that $M \le N/\log n$ if we choose first m polygons to be colored. To prove the claim, suppose $d_{m+1} \ge N/n$. Then $\sum_{j=m+1}^{n} d_j \ge (n - m) \cdot N/n = N(1 - 1/\log n)$, which implies $M = \sum_{j=1}^{m} d_j \le N/\log n$. Next, suppose $d_{m+1} < N/n$. In this case, $M \le N/n \cdot m = N/\log n$, which proves the claim. Thus, $\log P_{n,k,\bar{n}} \ge \sum_{i=2}^{k} n_i \cdot i \log \frac{n}{i} - n \log n - \mathrm{O}(N \log\log n)$.

Note that in our lower bound proofs we count only the number of intersection graphs on a circle which has a point not contained in any polygon, and therefore it also holds for interval and permutation graphs.

3.2 A Succinct Representation

Now we provide a succinct representation for generalized circle polygon graph G with n generalized polygons on a circle. Let N be the total number of corners of the polygons.

Note that the recognition algorithm of general k-polygon-circle graphs, which is a sub-class of the generalized circle polygon graphs, is NP-complete [20]. Thus we assume that G is given as a polygon-circle representation with n polygons, which is defined (for a graph $G = (V, E)$) as a mapping $\mathcal{P}$ of vertices in V to polygons inscribed into a circle such that $(u, v) \in E$ if and only if $\mathcal{P}(u)$ intersects $\mathcal{P}(v)$.

Then, a *corner-string* of a polygon-circle representation is a string produced by starting at any arbitrary location on the circle, and proceeding around the circle in clockwise order, adding a label denoting the vertex represented by a polygon each time a corner of a polygon encountered (denoted by the array S in Figure 1). Note that a single polygon-circle representation has many possible corner-strings, depending on the starting point. As the naive encoding of S uses $N\lceil \log n \rceil$ bits, it is not succinct, and does not support efficient queries. Therefore we convert S into another representation and add auxiliary data structures for efficient queries. First, we convert S into a bit array F of length N and another integer array S' of length $N-n$. The entry $F[i]$ is 1 if $S[i]$ is the first occurrence of the value in S, and 0 otherwise. The array S' stores all entries of S except for the first occurrence of each value in the same order as in S. We store F using the data structure of Lemma 2, and S' using the data structure of Lemma 3. Then the space becomes $(N-n)\log n + \mathrm{O}(N \log n/\log\log n))$ bits, which is succinct. Using F and S', we show how to support $\mathsf{access}(i, S)$ and $\mathsf{rank}_\alpha(i, S)$ in $\mathrm{O}(\log\log n)$ time and $\mathsf{select}_\alpha(i, S)$ in O(1) time.

- $\mathsf{access}(i, S)$
$$= \begin{cases} \mathsf{rank}_1(i, F) & (\mathsf{access}(i, F) = 1) \\ \mathsf{access}(\mathsf{rank}_0(i, F), S') & (\text{otherwise}) \end{cases}$$
- $\mathsf{rank}_\alpha(i, S)$
$$= \mathsf{rank}_\alpha(\mathsf{rank}_0(i, F), S') + \begin{cases} 1 & (\mathsf{rank}_1(i, F) \geq S[i]) \\ 0 & (\text{otherwise}) \end{cases}$$
- $\mathsf{select}_\alpha(i, S)$
$$= \begin{cases} \mathsf{select}_1(\alpha, F) & (i = 1) \\ \mathsf{select}_0(\mathsf{select}_\alpha(i-1, S'), F) & (\text{otherwise}) \end{cases}$$

We can regard as if the array S were stored and access to S were done in $\mathrm{O}(\log\log n)$ time. Therefore below we describe our algorithms using S.

We prove the space bound of Theorem 1.2. We compress the corner-string S, in which a character $2 \leq i \leq k$ appears n_i times in S. The length of S is $N = \sum_{i=2}^{k} n_i \cdot i$. For each character i, its first occurrence in S is encoded in a bit-vector of length N. Other characters are stored in a string S' of length $N - n$. Each character i appears $n_i - 1$ times in S'. We compress S' into its order-0 entropy. Then the total space is

$$\begin{aligned} \sum_{i=2}^{k} n_i(i-1)\log\frac{N-n}{i-1} + \mathrm{O}(N) &\leq \sum_{i=2}^{k} n_i(i-1)\log\frac{nk}{i/2} + \mathrm{O}(N) \\ &\leq \sum_{i=2}^{k} n_i \cdot i \log\frac{n}{i} - n\log n \\ &\quad + \mathrm{O}(N\log k). \end{aligned}$$

If $k = \mathrm{o}(\log n/\log\log n)$, the lower bound of Theorem 1.1 is

$$\begin{aligned} \log P_{n,k,\bar{n}} &\geq \sum_{i=2}^{k} n_i \cdot i \log\frac{n}{i} - n\log n - \mathrm{O}(N\log\log n) \\ &\geq \sum_{i=2}^{k} n_i \cdot i \log\frac{n}{i} - n\log n - \mathrm{o}(n\log n). \end{aligned}$$

On the other hand, the upper bound is

$$\begin{aligned} &\sum_{i=2}^{k} n_i \cdot i \log\frac{n}{i} - n\log n + \mathrm{O}(N\log k) \\ &\leq \sum_{i=2}^{k} n_i \cdot i \log\frac{n}{i} - n\log n + \mathrm{o}(n\log n). \end{aligned}$$

Therefore this upper bound matches the lower bound.

3.3 Query Algorithms

Our basic idea for queries is as follows. Consider a vertex u in G. Assume the vertex u corresponds to a k-polygon, which is represented by k many integers u in S. The polygon has k edges, for i-th edge ($1 \leq i \leq k-1$), we consider an interval of S between i-th occurrence of u and $(i+1)$-st occurrence of u, that is, $[\mathsf{select}_u(i, S), \mathsf{select}_u(i+1, S)]$. Let $I(u, i)$ denote this interval. For k-th edge, the interval becomes the union of $[\mathsf{select}_u(k, S), N]$ and $[1, \mathsf{select}_u(1, S)]$.

Consider two polygons u and v. We check for each side e of u if e intersects with a side f of v. Let $I(u, i) = [\ell, r]$ be the interval of e and $I(v, j) = [s, t]$ be the interval of f. There are four cases. (1) e is a chord and f is a chord. Then e and f intersect iff $[\ell, r] \cap [s, t] \neq \emptyset$, $[s, t] \not\subset [\ell, r]$, and $[\ell, r] \not\subset [s, t]$. (2) e is an arc and f is a chord. This case is the same as (1) in addition the case when $[s, t] \subset [\ell, r]$. (3) e is a chord and f is an arc. This case is the same as (1) in addition to the case when $[\ell, r] \subset [s, t]$. (4) e is an arc and f is an arc. Then e and f intersect iff $[\ell, r] \cap [s, t] \neq \emptyset$.

We add new data structures N, N_a, P_a, N_c, P_c, and A defined as follows. Let $I(u,i) = [\ell_i, r_i]$ denote the i-th interval of S, defined above, and d_u be the number of corners of u. Then A is a bit array of length N where $A[\ell_i] = 1$ if and only if $[\ell_i, r_i]$ corresponds to an arc of u. The arrays N, N_a, P_a, N_c, and P_c are defined as follows where $u = S[i]$.

$$N[i] = \begin{cases} \mathsf{select}_u(\mathsf{rank}_u(i,S)+1,S) & (\text{if } \mathsf{rank}_u(i,S) < d_u) \\ \infty & (\text{otherwise}) \end{cases}$$

$$N_a[i] = \begin{cases} \mathsf{select}_u(\mathsf{rank}_u(i,S)+1,S) & \left(\begin{array}{l}\text{if } A[i]=1 \text{ and} \\ \mathsf{rank}_u(i,S) < d_u\end{array}\right) \\ \infty & \left(\begin{array}{l}\text{if } A[i]=1 \text{ and} \\ \mathsf{rank}_u(i,S) = d_u\end{array}\right) \\ 0 & (\text{otherwise}) \end{cases}$$

$$P_a[i] = \begin{cases} \mathsf{select}_u(\mathsf{rank}_u(i,S)-1,S) & \left(\begin{array}{l}\text{if the side ending at} \\ S[i] \text{ is an arc and} \\ \mathsf{rank}_u(i,S) > 1\end{array}\right) \\ 0 & \left(\begin{array}{l}\text{if the side ending at} \\ S[i] \text{ is an arc and} \\ \mathsf{rank}_u(i,S) = 1\end{array}\right) \\ \infty & (\text{otherwise}) \end{cases}$$

$$N_c[i] = \begin{cases} \mathsf{select}_u(\mathsf{rank}_u(i,S)+1,S) & \left(\begin{array}{l}\text{if } A[i]=0 \text{ and} \\ \mathsf{rank}_u(i,S) < d_u\end{array}\right) \\ 0 & (\text{otherwise}) \end{cases}$$

$$P_c[i] = \begin{cases} \mathsf{select}_u(\mathsf{rank}_u(i,S)-1,S) & \left(\begin{array}{l}\text{if the side ending at} \\ S[i] \text{ is a chord and} \\ \mathsf{rank}_u(i,S) > 1\end{array}\right) \\ \infty & (\text{otherwise}) \end{cases}$$

The array $N_a[i]$ ($N_c[i]$) stores the other endpoint of an arc (a chord) starting from $S[i]$. The difference between N_a and N_c is that in N_c, we do not store the last side of a polygon. We do not store these arrays explicitly; we store only the range maximum data structures for N, N_a and N_c, and the range minimum data structure for P_a and P_c. We can obtain any entry of the arrays in O($\log\log n$) time using the above formula. This completes the proof.

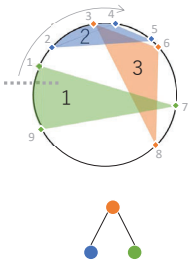
Lemma 5. *If generalized polygons u and v intersect, there exists a side e of u with interval $[\ell, r]$ and a side f of v with interval $[s,t]$ satisfying at least one of the following.*

(1) Both e and f are chords, neither e or f is the last side, and ($\ell < s < r$ and $r < t(= N_c[s])$) or ($\ell < t < r$ and $\ell > s(= P_c[t])$).

(2) e is an arc and f is a chord, f is not the last side, and $\ell < t(= N_a[s]) < r$ or $\ell < s(= P_a[t]) < r$.

(3) e is a chord and f is an arc, e is not the last side, and $s < r < t(= N_a[s])$ or $s(= P_a[t]) < \ell < t$.

(4) Both e and f are arcs, and $s < r$ and $\ell < t(= N_a[s])$.



$S = 1\,2\,3\,2\,2\,2\,3\,1\,3\,1$
$F = 1\,1\,1\,0\,0\,0\,0\,0\,0\,0$
$S' = 2\,2\,3\,1\,3\,1$
$A = 0\,1\,0\,0\,0\,0\,0\,0\,0\,1$
$N = 7\,4\,6\,5\,\infty\,8\,9\,\infty\,\infty$
$N_a = 0\,4\,0\,0\,0\,0\,0\,0\,0\,\infty$
$P_a = 0\,\infty\,\infty\,2\,\infty\,\infty\,\infty\,\infty\,\infty$
$N_c = 7\,0\,6\,5\,0\,8\,9\,0\,0$
$P_c = \infty\,\infty\,\infty\,\infty\,\infty\,4\,3\,1\,6\,7$

Fig. 1: A circle-trapezoid graph with $n = 3$ given as polygon-circle representation, and graph form. The array S encodes the corner-string starting from the marked location s on the circle. The arrays S, N_a, P_a, N_c, P_c are not stored; we store only F, S', A, range max data structures for N, N_a, N_c, and range min data structures for P_a, P_c.

Note that for an arc e that is the last side of u, the interval is divided into two. We regard as if e is divided into two arcs and apply the lemma to each of them.

Figure 1 shows an example of our representation. For an arc of polygon 2 whose interval is $[2,4]$, polygons 3 intersect with the chord because $N_c[3] = 5 > 4$. For a chord of polygon 3 whose interval is $[6,8]$, polygon 1 intersects with chords because $P_c[7] = 1 < 6$ and $N_c[7] = 9 > 8$.

Using this idea, we obtain an algorithm for adjacent query.

adjacent(u,v) query: Consider the intervals $I(u,1)$, $I(u,2), \ldots, I(u,d_u)$ and $I(v,1)$, $I(v,2), \ldots, I(v,d_v)$. We scan these intervals in the clockwise order on the circle, and for each endpoint of an interval, we check the condition of Lemma 5. For each interval, checking this condition takes $O(\log\log n)$ time, and since we need to check at most $d_u + d_v$ intervals, the time complexity is $O(k \log\log n)$.

Next we consider neighborhood(u) query. For each side (chord or arc) of u, we want to enumerate all generalized polygons v satisfying the conditions of Lemma 5. For each chord e of u, we can find all chords which intersects with u as follows. Let $[\ell, r]$ be the interval of e. First we obtain $m = \mathsf{rMq}(N_c, \ell, r)$. If $N_c[m] \le r$, all entries of N_c in $[\ell, r]$ are less than r, and there are no polygons intersecting e. Therefore we stop enumeration. If $N_c[m] > r$, the polygon $S[m]$ intersects with u. To check if there

is another such polygon, we recursively search for $[\ell, m-1]$ and $[m+1, r]$. The time complexity is $O(d \log\log n)$ where d is the number of entries m such that $N_c[m] > r$. We also process P_c analogously.

For an arc e of u, we can enumerate all chords of the other generalized polygons which intersects with e is obtained by finding all $S[m]$ such that $\ell < m < r$. Such distinct m can be obtained by finding all m such that (i) $\ell < m < r$, and (ii) $\ell < N_c[m]$ or $P_c[m] < r$ using the range maximum data structure.

For a chord e of u, we can enumerate all arcs of other generalized polygons which intersects with e is obtained by finding all $S[m]$ such that $1 \le m < r$ and $N_a[m] > r$, or $\ell < m \le N$ and $P_a[m] < \ell$.

For an arc e of u, we can enumerate all arcs of other generalized polygons which intersects with e is obtained by finding all $S[m]$ such that $1 \le m < r$ and $N_a[m] > \ell$, or $\ell < m \le N$ and $P_a[m] < r$.

neighborhood(u) query: For each interval $I(u,i) = [\ell, r]$, we output all polygons $S[m]$ satisfying one of the above conditions. However there may exist duplicates. To avoid outputting the same polygon twice, we use a bit array $D[1 \ldots, n]$ to mark which polygon is already output. The bit array is initialized by 0 when we create the data structure. At a query process, before outputting a polygon v, we check if $D[v] = 1$. If it is, v is already output and we do not output again. If not, we output v and set $D[v] = 1$. After processing all intervals of u, we have to clean D. To do so, we run the same algorithm again. But this time we output nothing and set $D[v] = 0$ for all v found by the algorithm. The time complexity is $O(|\mathsf{degree}(v)| \cdot k \log\log n)$ where k is the maximum number of sides in each generalized polygon.

degree(v) query: The degree(v) can be answered by returning the size of the output of the neighborhood(u) query, in $O(|\mathsf{degree}(v)| \cdot k \log\log n)$ time. Note that by adding an integer array of length n storing the degree of each vertex explicitly, degree(v) can be supported in $O(1)$ time. The whole data structure is still succinct if $k = \omega(1)$, but it is not if $k = O(1)$.

Finally, we show how one can represent various classes of intersection graphs by our representation. Generalized polygons in each class are represented as follows.

- k-polygon-circle: set all $A[i] = 0$ for all i (all sides are chords).
- circle-trapezoid: the number of sides is 4 and arcs and chords appear alternately.
- trapezoid: we split a circle in half equally (upper and lower part), and both upper and lower part have 2 corners. Now arcs and chords appear alternately, from the arcs on the upper part.
- circle and permutation: the number of sides is 2 and all sides are chords.
- circular-arc and interval: the number of sides is 2 and there are an arc and a chord. Set all the entries of N_c to be 0 and P_c to be ∞ so that the query algorithms do not output any chord.

Note that for circle and trapezoid graphs, we have alternative succinct representations which can answer degree(v) queries independent of $|\mathsf{degree}(v)|$, but takes more time for the other two queries compared to the representation of Theorem 1.2.

4. Conclusion Remarks

In this article we proved a unified space lower bound for several classes of intersection graphs on a circle. Subsequently, we designed succinct navigational oracles for these classes of graphs in a uniform manner, along with efficient support for queries such as degree, adjacency and neighborhood. We conclude with the following an open problem: can we improve the query times of our data structures, possibly to constant time?

References

[1] H. Acan, S. Chakraborty, S. Jo, and S. R. Satti. Succinct data structures for families of interval graphs. In *WADS*, volume 11646 of *LNCS*, pages 1–13. Springer, 2019.
[2] L. C. Aleardi, O. Devillers, and G. Schaeffer. Succinct representations of planar maps. *Theor. Comput. Sci.*, 408(2-3):174–187, 2008.
[3] J. Barbay, F. Claude, T. Gagie, G. Navarro, and Y. Nekrich. Efficient fully-compressed sequence representations. *Algorithmica*, 69(1):232–268, 2014.
[4] F. Bazzaro and C. Gavoille. Localized and compact data-structure for comparability graphs. *Discrete Mathematics*, 309(11):3465–3484, 2009.
[5] A. Bouchet. Characterizing and recognizing circle graphs. Graph theory, Proc. 6th Yugosl. Semin., 1986.
[6] D. R. Clark and J. I Munro. Efficient suffix trees on secondary storage. SODA '96, pages 383–391, 1996.
[7] J. Enright and S. Kitaev. Polygon-circle and word-representable graphs. *Electronic Notes in Discrete Mathematics*, 71:3–8, 2019.
[8] S. Even and A. Itai. Queues, stacks and graphs. *Theory of Machines and Computations*, pages 71–86, 1971.
[9] A. Farzan and S. Kamali. Compact navigation and distance oracles for graphs with small treewidth. *Algorithmica*, 69(1):92–116, 2014.
[10] A. Farzan and J. I. Munro. Succinct encoding of arbitrary graphs. *Theor. Comput. Sci.*, 513:38–52, 2013.
[11] S. Felsner, R. Müller, and L. Wernisch. Trapezoid graphs and generalizations, geometry and algorithms. *Discrete Applied Mathematics*, 74(1):13–32, 1997.
[12] J. Fischer and V. Heun. Space-efficient preprocessing schemes for range minimum queries on static arrays. *SIAM J. Comput.*, 40(2):465–492, 2011.
[13] C. Gavoille and C. Paul. Optimal distance labeling for interval graphs and related graph families. *SIAM J. Discrete Math.*, 22(3):1239–1258, 2008.
[14] M. C. Golumbic. Interval graphs and related topics. *Discrete Mathematics*, 55(2):113–121, 1985.
[15] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 2004.
[16] M. Habib, R. M. McConnell, C. Paul, and L. Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theor. Comput.*

Sci., 234(1-2):59–84, 2000.
[17] G. Hajós. Über eine art von graphen. *Int. Math. Nachr.*, 11:1607–1620, 1957.
[18] M. Koebe. On a new class of intersection graphs. *Ann. Discrete Mathematics*, pages 141–143, 1992.
[19] Y. Koh and S. Ree. Connected permutation graphs. *Discrete Mathematics*, 307(21):2628–2635, 2007.
[20] J. Kratochvíl and M. Pergel. Two results on intersection graphs of polygons. In *GD*, pages 59–70, 2003.
[21] T. A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory*. SIAM Monographs on Discrete Mathematics and Applications, 1999.
[22] J. I. Munro and V. Raman. Succinct representation of balanced parentheses and static trees. *SIAM J. Comput.*, 31(3):762–776, 2001.
[23] J. I. Munro and K. Wu. Succinct data structures for chordal graphs. In *ISAAC*, pages 67:1–67:12, 2018.
[24] G. Navarro. *Compact Data Structures - A Practical Approach*. Cambridge University Press, 2016.
[25] M. Pergel. Recognition of polygon-circle graphs and graphs of interval filaments is NP-complete. In *WG*, pages 238–247, 2007.
[26] J. P. Spinrad. *Efficient graph representations*, volume 19 of *Fields Institute monographs*. American Mathematical Society, 2003.